

VAX/VMS DBOYLE SPRITES 9-DEC-1983 08:51 LCA0: 9-DEC-1983 09:09 DISK\$VAXVMSRL3:[USER.DBOYLE]SPRITES.MEM;24 VAX/VMS
VAX/VMS DBOYLE SPRITES 9-DEC-1983 08:51 LCA0: 9-DEC-1983 09:09 DISK\$VAXVMSRL3:[USER.DBOYLE]SPRITES.MEM;24 VAX/VMS
VAX/VMS DBOYLE SPRITES 9-DEC-1983 08:51 LCA0: 9-DEC-1983 09:09 DISK\$VAXVMSRL3:[USER.DBOYLE]SPRITES.MEM;24 VAX/VMS

TIMEX COMPUTER CORPORATION

APPLICATION DEVELOPMENT LIBRARY

Application Software Component

Date: 09 Dec 83

Version: 001

ASC Number: XXX

Author: David Boyle

Name: Sprite Graphics Support

Description: This component provides support for sprite graphics. A sprite is a graphical object which can be created and manipulated by a set of services to be described below. A sprite can be up to 255 x 255 characters in size, however only 32 x 22 can be displayed. Each sprite is identified by a sprite-id, a number greater or equal to zero, and has the following properties:

definition - a pointer to a width x height character bitmap, where width and height are user defined. A 1 indicates foreground color, a 0 indicates background color.

position - the row, col location on the screen of the upper left corner of the sprite.

color - the screen attribute (foreground color). A value of 0 indicates black, 1 indicates blue, etc.

size - the width and height of the sprite (in characters).

There is a variable, SPRITES at 5EE8H, which contains the location of the sprites data structure. This variable must

be initialized before using any of the sprite services. The sprite support services can be invoked from machine code and BASIC. There is a BASIC interface routine, which takes as its parameter a BASIC string variable. The name of this variable is specified by poking in the starting character into the location pointed to by SPRITES. For example, if `POKE 24296, CODE("C")` is executed, the variable `C$` will be used to pass commands to the interface routine. This must be done before using any sprite services. The interface routine is invoked by `LET <variable> = USR <TBS>`. The status code returned by the last sprite service executed is assigned to `<variable>`.

The BASIC interface routine will cause report "A: Invalid argument" to be produced whenever an illegal command or invalid argument to a command is found. If too few or too many arguments are given for some command, the report "Q: Parameter error" is produced. The machine code routines will all return status codes in the BC register pair. If the value in C is 0, then no error occurred. In this case B contains a value indicating further status information. If an error occurred, C will contain an odd value (thus bit 0 is set).

The BASIC interface string variable contains sprite graphics commands, separated by spaces. The commands have the following syntax:

"<command letter><parameter list>"

where <command letter> is a single upper or lower case letter identifying the command:

C = Create_Sprite
 S = Set_Autowrap_Mode
 P = Put_Sprite
 E = Erase_Sprite
 M = Move_Sprite
 L = Change_Sprite_Location
 A = Change_Sprite_Attribute
 O = Overlap?
 H = Horizontal_Scroll
 V = Vertical_Scroll

and where <parameter list> is a list of numbers separated by commas. The string assigned to `C$` can be a string expression, i.e., "`<substr>+VAL$(<expr>)+<substr>`". The semantics of these commands will be described in detail below.

ASC Number: XXX

Application Services

Name: Create_Sprite

Machine code interface:

Inputs: A = sprite-id, 0..255
 B = width, 0..255
 C = height, 0..255
 D = color, 0..7
 HL = definition address, 0..64K-1

Outputs: BC = 00h - OK (sprite created)
 = 01h - not enough memory

BASIC interface:

Inputs: LET C\$ =
 "C<sprite-id>,<width>,<height>,<color>,<addr>"

Description: This service creates a sprite with the specified width, height, color, and definition address. Initially, the position of the sprite is (0, 0). If there was already a sprite with the specified id, it is destroyed. In all further operations the new sprite is identified by its sprite-id. If there is not enough free memory for the new sprite, a 01h is returned, otherwise a 00h is returned.

BASIC example: "C0,2,2,2,32768" creates sprite 0 which is 2 x 2 characters in size, is red in color, and whose definition is located at 32768. The initial position of the sprite is (0, 0).

Name: Set_Autowrap_Mode

Machine code interface:

Inputs: A = mode: 0 - off, not 0 - on

Outputs: BC = 00h - OK

BASIC interface:

Inputs: LET C\$ = "S<mode>"

Description: This service turns the Autowrap mode on and off. When the Autowrap mode is on, all changes to a sprite's location are made so that the new location is on the screen (wraparound).

BASIC example: "S1" causes Autowrap mode to be turned on.

Name: Put_Sprite

Machine code interface:

Inputs: A = sprite-id, 0..255
 D = row, 0..255
 E = column, 0..255

Outputs: BC = 00h - OK, nothing overwritten
 10h - OK, something overwritten
 03h - illegal sprite-id
 D = row
 E = column

BASIC interface:

Inputs: LET C\$ = "P<sprite-id>,<row>,<column>"

Description: This service writes a sprite on the screen. The row and column described above define the location at which the upper left corner of the sprite is written. If the Autowrap mode is set, the row and column values are modified to make sure that the sprite location is a legal screen location. The new row and column values are returned. If the Autowrap mode is not set, the position is not changed. However, only those parts of the sprite corresponding to legal screen positions will be displayed. If writing the sprite causes something on the screen to be overwritten, the appropriate code is returned.

BASIC example: "P0,11,0" puts sprite 0 at row 11, column 0. Anything already at that location is overwritten. If anything is overwritten, the interface routine returns 10h.

Name: Erase_Sprite

Machine code interface:

Inputs: A = sprite-id, 0..255

Outputs: BC = 00h - OK (sprite erased)
 03h = illegal sprite-id

BASIC interface:

Inputs: LET C\$ = "E<sprite-id>"

Description: This service erases a sprite. The screen becomes blank where the sprite used to be. The sprite still exists and can be written elsewhere.

BASIC example: "E0" erases sprite 0. The screen locations taken up by the sprite become blank.

Name: Move_Sprite

Machine code interface:

Inputs: A = sprite-id, 0..255
 D = relative horizontal motion, -128..127
 E = relative vertical motion, -128..127

Outputs: BC = 00h - OK, nothing overwritten
 10h - OK, something overwritten
 03h - illegal sprite-id
 D = row
 E = column

BASIC interface:

Inputs: LET C\$ = "M<sprite-id>,<hor>,<vert>"

Description: This service moves a sprite. The sprite is erased at its current location and written at a new location defined by:

new row = old row + relative vertical motion
 new col = old col + relative horizontal motion

The location of the sprite is updated to reflect the motion. As with the "P" command, if the sprite overwrites anything, 10h is returned. If Autowrap mode is set, then the location is automatically wrapped to fit onto the screen (i.e., (32, 0) becomes (0, 0)). The new row and column values are returned. If Autowrap mode is not set, only those parts of the sprite which correspond to legal screen locations will be displayed.

BASIC example: "M0,2,0" changes the position of the sprite from (11, 0) to (13, 0). This operation is equivalent to "E0 P0,13,0".

Name: Change_Sprite_Location

Machine code interface:

Inputs: A = sprite-id, 0..255
D = new row, 0..255
E = new column, 0..255

Outputs: BC = 00h - OK (location changed)
03h - illegal sprite-id

BASIC interface:

Inputs: LET C\$ = "L<sprite-id>,<new-row>,<newcol>"

Description: This service updates the position property of a sprite in the same fashion as for Put_Sprite. The sprite is not updated on the screen until a Move_Sprite, Put_Sprite, or Erase_Sprite is executed.

BASIC example: "L0,13,0" will set the location of sprite 0 to (13, 0).

Name: Change_Sprite_Attribute

Machine code interface:

Inputs: A = sprite-id, 0..255
D = color, 0..7

Outputs: BC = 00h - OK (color changed)
03h - illegal sprite-id
05h - illegal color

BASIC interface:

Inputs: LET C\$ = "A<sprite-id>,<color>"

Description: This service updates the color property of a sprite. The sprite on the screen is not updated until a Put_Sprite, Move_Sprite, or Erase_Sprite is executed.

BASIC example: "A0,2" causes the color property of sprite 0 to be set to red.

Name: Overlap?

Machine code interface:

Inputs: D = sprite-id-1, 0..255
E = sprite-id-2, 0..255

Outputs: BC = 00h - OK, no overlap
20h - OK, overlap
03h - illegal sprite-id

BASIC interface:

Inputs: LET C\$ = "0<sprite-id-1>,<sprite-id-2>"

Description: This service is used to detect if two sprites overlap.

BASIC example: "00,1" causes the interface routine to return TBS if sprites 0 and 1 overlap on the screen.

Name: Horizontal_Scroll

Machine code interface:

Inputs: A = direction: positive number - right,
negative number - left

Outputs: BC = 00h - DK

BASIC interface:

Inputs: LET C\$ = "H<direction>"

Description: This service is used to scroll the entire screen in the horizontal direction by one column. The direction of scroll is to the left if direction is less than 0, otherwise the scroll is to the right. The position property of all sprites is updated by one column to reflect the effect of the scroll. If Autowrap mode is not set, then 1 is added to the column value. If Autowrap mode is set, then 1 is added to the column value, and this new value is wrapped to fit on the screen.

BASIC example: "H1" causes the screen to be scrolled to the right one column.

Name: Vertical_Scroll

Machine code interface:

Inputs: A = direction: positive number - down,
 negative number - up

Outputs: BC = 00h - OK

BASIC interface:

Inputs: LET C\$ = "V<direction>"

Description: This service is used to scroll the entire screen in the vertical direction by one column. The direction of scroll is up if direction is less than 0, otherwise the scroll is down. The position property of all sprites is updated by one row to reflect the effect of the scroll. If Autowrap mode is not set, then 1 is added to the column value. If Autowrap mode is set, then 1 is added to the column value, and this new value is wrapped to fit on the screen.

BASIC example: "V1" causes the screen to scrolled down one row.
